

[51]Int. Cl<sup>6</sup>

G06F 9/38

## [12] 发明专利申请公开说明书

[21] 申请号 98115530.8

[43]公开日 1999 年 1 月 27 日

[11]公开号 CN 1206145A

[22]申请日 98.6.29 [21]申请号 98115530.8

### [30] 优先权

[32]97.6.30 [33]]P [31]174407/97

[71] 申请人 索尼公司

地址 日本东京都

[72]发明人 华木博一

[74] 专利代理机构 中国专利代理(香港)有限公司

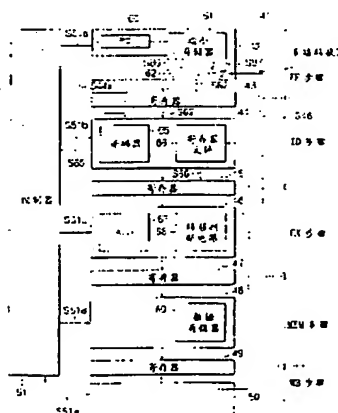
代理人 程天正 陈景峻

权利要求书 3 页 说明书 11 页 附图页数 6 页

[54]发明名称 带有流水线处理电路的信号处理器及其方法

**[57]摘要**

一种用于流水线处理的信号处理器及方法,该处理器能有效地避免因转移指令而导致的处理效率下降,其中当获得了关于 ID 模块内所译码的指令是转移指令的结果时,就在下一个周期内对 EX 模块内的转移存在情况进行判断,并在 IF 模块内同时取出转移目的地的指令和非转移目的地的指令,随后在下一周期中,根据存在转移的结果选定所取出的转移目的地或非转移目的地的指令,然后在 ID 模块内对该指令译码。



# 权 利 要 求 书

1. 一种微处理器，它包括：  
用于存储指令的装置；  
用于从上述指令存储装置中取出指令的装置；  
5 用于对取出的指令进行译码的装置；  
用于执行上述译码后的指令的装置；  
一存储器；  
用于访问上述存储器的装置；  
用于将所执行的结果写入被访问的存储器的装置；以及，  
10 用于对指令读取装置、指令译码装置、指令执行装置、存储器访问装置以及写入装置中的操作进行流水线处理的装置。

所述指令读取装置包括：

- 一程序计数器，它以顺序的方式指定指令存储装置中的地址；  
一地址存储部分，它在译码后的指令是转移指令时存储一地址，该  
15 地址是被包括在转移指令中的转移目的地的地址；  
一指令存储部分，它带有多个其中存储有指令的可同时访问的存储区；

- 一读取部分，它可在译码后的指令是转移指令时同时取出存储在指令存储装置内第一地址处的第一指令以及存储在指令存储装置内第二地址处的第二指令，所述第一地址由程序计数器来指示，所述第二地址由  
20 存储在地址存储装置内的地址来指示；以及，

一选择部分，它用于根据对转移指令中的转移条件的判断而选择同时取出的第一和第二指令中的一个并将该指令输出给指令译码装置。

2. 如权利要求 1 的微处理器，其特征在于，所述指令存储部分将一  
25 转移指令的转移目的地的指令以及该转移指令的非转移目的地的另一条指令存储到不同的存储区内。

3. 如权利要求 1 的微处理器，其特征在于，所述指令存储部分将对应于存储区数目的多个顺序且连续处理的指令存储到不同的存储区。

4. 如权利要求 1 的微处理器，其特征在于，所述指令存储部分包括  
30 一单端口式存储器，它具有一个单个的读出端口。

5. 如权利要求 1 的微处理器，其特征在于，所述指令读取装置还包括：

一标志存储部分，该部分存储表示存储在地址存储部分内的地址合法性的标志，并且，

该指令读取装置仅在存储在标志存储部分内的标志表示地址是合法的时才取出存储在指令存储部分内的地址，该地址由存储在地址存储部分内的地址来指示。

6. 如权利要求 1 的微处理器，其特征在于，所述指令读取装置内的读取部分根据地址的第一部分来指定存储区，并根据地址的第二部分指定存储区内的地址。

7. 如权利要求 1 的微处理器，其特征在于，所述指令译码装置包括：  
一译码部分，它用于对在选择部分处选定的指令进行译码并生成一控制信号以便执行译码后的指令；以及，

一数据存储部分，它存储指令执行装置中所使用的数据。

8. 如权利要求 1 的微处理器，其特征在于，所述指令执行装置包括：  
一算法与逻辑处理部分，以及  
一转移判断部分，该部分用于判断转移指令的转移条件。

9. 如权利要求 1 的微处理器，其特征在于，所述写入装置将指令执行装置的处理结果存储进存储器和指令译码装置中的数据存储部分。

10. 如权利要求 1 的微处理器，所述微处理器包括：

一单个指令读取装置；  
一单个指令译码装置；  
一单个指令执行装置；  
一单个指令访问装置以及  
一单个写入装置。

11. 一种处理信号的方法，该方法包括下列步骤：

从指令存储装置中读取出一条指令；

对取出的指令进行译码；

执行译码后的指令；

访问存储器；

将执行后的结果写入所访问的存储器；以及，

以流水线的方式处理所说的读取、译码、执行、访问和写入操作，  
所述读取步骤包括下列步骤：

顺序地指定指令存储装置中的地址以指明一非转移目的地的指令的

地址;

在译码后的指令是转移指令时存储一地址, 该地址是被包括在转移指令中的转移目的地地址;

- 5 将转移目的地的指令和非转移目的地的另一条指令存入指令存储装置中不同的可同时访问的存储区内;

当译码后的指令是一转移指令时, 同时读取存储在指令存储装置中第一地址处的第一指令和存储在指令存储装置中第二地址处的第二指令, 所述第一地址由程序计数器来指示, 所述第二地址由存储在地址存储装置中的地址来指示;

- 10 根据对转移指令的转移条件的判断而选择出同时取出的第一和第二指令中的一个; 以及,

将所选定的取出指令输出给指令译码装置。

# 说明书

## 带有流水线处理电路的信号

### 处理器及其方法

5 本发明涉及带有流水线电路的信号处理器及其方法。

安装在数字信号处理器 (DSP) 内的精简指令集计算机 (RISC) 等通常如下所述那样根据程序进行信号处理。也就是说, 一个处理器通过顺序地执行下列指令步骤 (阶段) 来对程序中的各条指令进行信号处理: 指令读取步骤 (IF 步骤), 它用于从指令存储器中取出指令; 指令译码  
10 步骤 (ID 步骤), 它用于对取出的指令进行译码; 执行步骤 (EX 步骤), 它用于执行已译码的指令; 存储器访问步骤 (MEM 步骤), 它用于访问存储器; 以及, 写入步骤 (WB 步骤), 它用于将上述访问所获得的结果写入存储器。

在这种情况下, 在把用于读取指令的时间调整到用于前一指令的 WB  
15 步骤结束之后时间时, 对各个 IF 步骤、ID 步骤、EX 步骤、MEM 步骤以及 WB 步骤来说, 从开始读取前一指令的时间到用于下一指令的 WB 步骤结束时间需要花费双倍的总时间。

图 1 是相关技术的计算机处理器 1 的框图。

如图 1 所示, 处理器 1 包括一 IF 模块 2、一寄存器 3、一 ID 模块 4、  
20 一寄存器 5、一 EX 模块 6、一寄存器 7、一 MEM 模块 8、一寄存器 9、一 WB 模块 10 以及一控制器 11。

IF 模块 2、ID 模块 4、EX 模块 6、MEM 模块 8、WB 模块 10 分别执行 IF 步骤、ID 步骤、EX 步骤、EX 步骤、MEM 步骤以及 WB 步骤。

这里, 在处理器 1 中, 为了增加单位时间的处理量, 通常采用流水  
25 线处理, 这种处理并行地执行上述不同步骤的处理。

在流水线处理中, 如图 2 所示, 在一个周期内完成所有步骤的处理, 就每个周期而言, 指令顺序地输入给处理器, 并且, 并行地执行有 IF 步骤、ID 步骤、EX 步骤、MEM 步骤和 WB 步骤的不同指令。

具体地说, 在图 1 所示的处理器 1 中, 在一个周期间隔将指令  $n$  至  
30  $n+4$  输入给处理器 1。在周期 20 中, 并行地执行指令  $n$  的 WB 步骤、指令  $n+1$  的 MEM 步骤、指令  $n+2$  的 EX 步骤、指令  $n+3$  的 IF 步骤以及指令  $n+4$  的 IF 步骤。

通过这种方式, 在使用五步流水线处理时, 与不用流水线处理的情况相比, 每个周期的处理量可增加四倍。

尽管参照使用五步流水线处理的实例说明了上述处理器 1, 但是, 也可以对指令的处理进一步细分以简化每一步骤中的处理, 从而提高时钟频率并增加单位时间的处理量。

如上所述, 在处理器 1 中, 如图 2 所示, 在开始指令  $n$  的 EX 步骤时, 就开始指令  $n+1$  的 ID 步骤和指令  $n+2$  的 IF 步骤。

当指令  $n$  是一转移指令时, 会在 ID 步骤中识别出指令  $n$  是否是转移指令。但是, 仅在处理 EX 步骤中的指令  $n$  时才判断是否转移, 也即是否满足转移条件。因此, 在确定了指令  $n$  是一转移指令时, 已经取出了指令  $n$  之后的指令  $n+1$  和  $n+2$ 。

这时, 如果指令  $n+1$  和  $n+2$  继续进入流水线处理, 就会终止执行用于非转移目的地的指令 (紧接着转移指令之后的指令), 从而不能正确地执行。

为了避免上述情况, 例如, 如图 3 所示, 当在 EX 步骤中确定出一条指令是转移指令时, 就中断业已取出的后续指令  $n+1$  和  $n+2$ , 并且, 顺序地读取下一个周期的转移目的地处的指令  $m$  和  $m+1$ 。

但是, 中断业已取出的指令具有降低处理效率的缺点。例如, 在图 3 所示的情况中, 转移会导致两个周期的延时。

为了克服上述现象, 使用了设置转移指令之后的指令的“延时转移”技术, 因此, 将以与判断转移指令是否存在无关的方式执行的指令放置在紧接着转移指令之后, 并且, 延时执行那些与是否有转移指令有关的指令。这里, 转移指令之后的指令中以与转移无关的方式执行的一组指令称为“延时段 (delay slot)”。

在使用上述延时转移技术时, 如果延时段中的指令数量大于取出之后因转移而中断的指令数量, 则可以把延时段放置于紧接在转移指令之后。如果不是这种情况, 必须将一指示系统什么都不做的“nop” (空操作) 指令放置于紧接在转移指令之后。因此, 存在处理效率下降的缺点。

还有诸如在 ID 步骤中识别出转移指令时使流水线停止、仅在转移判断之后取出一转移目的地或一非转移目的地指令、然后使流水线重新开始之类的其它方法。

但是，不论使用什么方法，都不可能在执行转移指令（转移判断）之前指定取下一条指令，所以，流水线会停止，直至指定读取的指令，从而处理效率会下降。

因此，使用流水线处理的处理器 1 具有由转移指令所导致的“转移损失”。不可能减少这种损失以便有更好的效率。

为了尽可能地减少这种转移损失，存在有事先预测转移的方法。但是，如果预测错误，则会导致较大的损失。还有，安装预测电路具有增加处理器尺寸的缺点。

另一种方法是在 ID 步骤中进行转移判断并立即进行转移。但是，如果转移指令之前的指令（在 EX 步骤中）正在处理上述判断所涉及的数据，则会出现临界的路径，难以进行高速安装。

本发明考虑到了上述相关技术。本发明的一个目的是提供一种用于流水线处理的信号处理器及其方法，所述信号处理器及方法能有效地抑制因转移指令所导致的处理效率下降。

依照本发明的第一个方面，提供了一种信号处理器，它包括：用于存储指令的装置；用于从上述指令存储装置中取出指令的装置；用于对取出的指令进行译码的装置；用于执行上述译码后的指令的装置；一存储器；用于访问上述存储器的装置；用于将所执行的结果写入被访问的存储器的装置；以及，用于对指令读取装置、指令译码装置、指令执行装置、存储器访问装置以及写入装置中的操作进行流水线处理的装置。所述指令读取装置包括：一程序计数器，它以顺序的方式指定指令存储装置中的地址；一地址存储部分，它在译码后的指令是转移指令时存储一地址，该地址是被包括在转移指令中的转移目的地的地址；一指令存储部分，它带有多个其中存储有指令的可同时访问的存储区；一读取部分，它可在译码后的指令是转移指令时同时取出存储在指令存储装置内第一地址处的第一指令以及存储在指令存储装置内第二地址处的第二指令，所述第一地址由程序计数器来指示，所述第二地址由存储在地址存储装置内的地址来指示；以及，一选择部分，它用于根据对转移指令中的转移条件的判断而选择同时取出的第一和第二指令中的一个并将该指令输出给指令译码装置。

最佳的是，指令存储部分将一转移指令的转移目的地的指令以及该转移指令的非转移目的地的另一条指令存储到不同的存储区内。

更佳的是，指令存储部分将对应于存储区数目的多个顺序且连续处理的指令存储到不同的存储区。

最佳的是，指令存储部分包括一单端口式存储器，它具有一个单个的读出端口。

5       最佳的是，指令读取装置还包括一标志存储部分，该部分存储表示存储在地址存储部分内的地址合法性的标志，并且，该指令读取装置仅在存储在标志存储部分内的标志表示地址是合法的时才取出存储在指令存储部分内的地址，该地址由存储在地址存储部分内的地址来指示。

10       最佳的是，指令读取装置内的读取部分根据地址的第一部分来指定存储区，并根据地址的第二部分指定存储区内的地址。

最佳的是，指令译码装置包括：一译码部分，它用于对在选择部分处选定的指令进行译码并生成一控制信号以便执行译码后的指令；以及，还包括一数据存储部分，它存储指令执行装置中所使用的数据。

15       最佳的是，指令执行装置包括一算法与逻辑处理部分以及一转移判断部分，该部分用于判断转移指令的转移条件。

最佳的是，写入装置将指令执行装置的处理结果存储进存储器和指令译码装置中的数据存储部分。

20       更佳的是，所述信号处理器包括一单个指令读取装置、一单个指令译码装置、一单个指令执行装置、一单个指令访问装置以及一单个写入装置。

依照本发明的第二个方面，提供了处理信号的方法，该方法包括下列步骤：从指令存储装置中读取出一条指令；对取出的指令进行译码；执行译码后的指令；访问存储器；将执行后的结果写入所访问的存储器；以及，以流水线的方式处理所说的读取、译码、执行、访问和写入操作，  
25       所述读取步骤包括下列步骤：顺序地指定指令存储装置中的地址以指明一非转移目的地指令的地址；在译码后的指令是转移指令时存储一地址，该地址是被包括在转移指令中的转移目的地地址；将转移目的地的指令和非转移目的地的另一条指令存入指令存储装置中不同的可同时访问的存储区内；当译码后的指令是一转移指令时，同时读取存储在指令  
30       存储装置中第一地址处的第一指令和存储在指令存储装置中第二地址处的第二指令，所述第一地址由程序计数器来指示，所述第二地址由存储在地址存储装置中的地址来指示；根据对转移指令的转移条件的判断而

选择出同时取出的第一和第二指令中的一个；以及，对所选定的取出指令进行译码。

以下参照附图详细说明本发明的上述和其它目的及特点，在附图中：

5 图 1 是相关技术的处理器的框图；

图 2 是图 1 所示处理器中流水线处理的示意图；

图 3 是在图 2 的流水线处理中执行一转移指令时的处理过程的示意图；

图 4 是本发明一个实施例的处理器的框图；

10 图 5 是图 4 所示指令存储器的框图；

图 6 是图 5 中存储器内指令存储格式的示意图；以及

图 7 是在图 4 所示的处理器中通过流水线处理来执行一转移指令时的处理过程的示意图。

以下说明本发明一个实施例的处理器。

15 图 4 是本发明实施例的处理器 41 的框图。

如图 4 所示，处理器 41 例如包括一模块 42、一寄存器 43、ID 模块 44、寄存器 45、EX 模块 46、寄存器 47、MEM 模块 48、寄存器 49、WB 模块 50 以及控制器 51。

20 IF 模块 42、ID 模块 44、EX 模块 46、MEM 模块 48 以及 WB 模块 50 分别执行 IF 步骤、ID 步骤、EX 步骤、MEM 步骤和 WB 步骤。

25 处理器 41 执行与上述处理器 1 相同的流水线处理。但以不同于处理器 1 的方式处理转移指令。也就是说，在处理器 41 中，以与图 1 中处理器 1 相同的方式在一个周期内完成对各个步骤的处理，指令在每个周期内顺序地输入给处理器，通过流水线处理并行地执行用于五个指令的 IF 步骤、ID 步骤、EX 步骤和 MEM 步骤。

但是，与处理器 1 不同，处理器 41 在 ID 模块中对指令进行译码，并在把一指令识别为转移指令时判断下一个周期的 EX 模块 46 中是否存在一个转移，且在 IF 模块 42 中同时取出用于转移目的地的指令和用于非转移目的地的指令。在随后的周期中，根据转移判断的结果选定取出的用于转移目的地或非转移目的地的指令中的一个，并在 ID 模块 44 中对选出的指令进行译码。

以下详细说明图 4 所示的处理器 41 的结构单元。

首先说明 IF 模块 42。

如图 4 所示, IF 模块 42 例如包括一程序计数器 60、指令存储器 61 以及用作一个选择单元的多路转换器 62。

5 程序计数器 60 根据来自控制器 51 的控制信号 S51a 指示下次要读取的指令在指令存储器 61 内的地址并在每一个周期均顺序地使该地址增一。

图 5 是指令存储器 61 的框图。

如图 5 所示, 指令存储器 61 包括用作指令存储单元的存储器 80、一标志寄存器 81、地址寄存器 82 和 83、访问控制单元 84<sub>1</sub> 至 84<sub>8</sub> 以及  
10 多路转换器 86 和 87。

存储器 80 是一单端口存储器, 它具有八个存储单元例如 80<sub>1</sub> 至 80<sub>8</sub>, 可同时访问这八个存储单元。通过把一单端口存储器用作存储器 80, 可以减小设备的尺寸并降低成本。

最佳的是, 将存储器 80 的存储单元的数量设置为 2 的幂。

15 如图 6 所示, 程序中的指令 1、2、3、4、5、6、7 和 8 顺序地存储在存储单元 80<sub>1</sub>、80<sub>2</sub>、80<sub>3</sub>、80<sub>4</sub>、80<sub>5</sub>、80<sub>6</sub>、80<sub>7</sub> 和 80<sub>8</sub> 中, 然后, 指令 9... 顺序地存储于存储单元 80<sub>1</sub> 到 80<sub>8</sub>。因此, 在一条转移指令出现时, 用于转移目的地的指令和用于非转移目的地的指令被存储到同一存储单元内的概率为八分之一。当用于转移目的地的指令和用于非转移目的地的指令存储在同一存储单元内时, 不可能同时取出这些指令。当这种情况出现时, 在不改变程序语义(意义)的情况下用另一条指令来替换这些指令中的一个。如果这种替换是不可能的, 则将“nop”(空操作)指令  
20 插入移位指令, 因此, 用于转移目的地的指令和用于非转移目的地的指令不会到达同一存储单元。

25 结果, 在转移指令出现时, 可以将用于转移目的地的指令和用于非转移目的地的指令存入不同的存储单元并能同时读出这些指令。

当存储器 80 具有如前所述那样的 8 个存储单元结构时, 存储在地  
址寄存器 82 和 83 中的地址的较低 3 位表示存储单元号, 较高位表示各存储单元中的地址。

30 当存储在地址寄存器 82 或 83 中的地址的较低三位分别是 000, 001, 010, 011, 100, 101, 110 和 111 时, 存储单元 80<sub>1</sub> 至 80<sub>8</sub> 被启动。

地址寄存器 82 将地址存入存储器 80, 存储器 80 则存储有由程序计

数器 60 所指示的用于非转移目的地的指令。

地址寄存器 83 将地址存入存储器 80，存储器 80 则存储有来自 ID 模块 44 的用于转移目的地输入的指令。

5 因此，IF 模块 42 有两个地址寄存器，它们用于同时访问存储器 80 的两个存储单元。

标志寄存器 81 存储有一标志，它指示存储在地址寄存器 83 内的转移目的地的地址是否合法。标志寄存器 81 在来自 ID 模块 44 的转移目的地的地址存入地址寄存器 83 时存储有标志〔1〕，而在其它情况下则存储有标志〔0〕。

10 多路转换器 85<sub>1</sub> 至 85<sub>8</sub> 例如根据来自控制器 51 的控制信号 S51a 来选定存储在地址寄存器 82 内的用于非转移目的地的指令地址和存储在地址寄存器 83 内的用于转移目的地的指令地址这二者中的一个，并将它们输出以便分别访问控制单元 84<sub>1</sub> 至 84<sub>8</sub>。

15 当来自多路转换器 85<sub>1</sub> 至 85<sub>8</sub> 的地址的较低三位表示相应的存储单元 80<sub>1</sub> 至 80<sub>8</sub> 时，访问控制单元 84<sub>1</sub> 至 84<sub>8</sub> 会根据所说的来自多路转换器 85<sub>1</sub> 到 85<sub>8</sub> 的地址用较高位分别从存储单元 80<sub>1</sub> 至 80<sub>8</sub> 中读出指令。

当存储在标志寄存器中的标志表示〔1〕时，访问控制单元 84<sub>1</sub> 至 84<sub>8</sub> 不会用存储在地址寄存器 83 中的地址对存储单元 80<sub>1</sub> 至 80<sub>8</sub> 进行读操作。

20 多路转换器 86 从自访问控制单元 84<sub>1</sub> 至 84<sub>8</sub> 中读取的结果之中选定从由存储在地址寄存器 82 中的地址的较低三位所指定的存储单元 80<sub>1</sub> 至 80<sub>8</sub> 中读出的结果，并将选定的转移目的地的指令 S86 输出给多路转换器 82。

25 多路转换器 86 从自访问控制单元 84<sub>1</sub> 至 84<sub>8</sub> 中读取的结果之中选定从由存储在地址寄存器 83 中的地址的较低三位所指定的存储单元 80<sub>1</sub> 至 80<sub>8</sub> 中读出的结果，并将选定的转移目的地的指令 S87 输出给多路转换器 62。

30 在 IF 模块 42 中，同时读出由存储在地址寄存器 82 中的地址所表示的非转移目的地的指令地址和由存储在地址寄存器 82 中的地址所表示的转移目的地的指令地址。这时，转移指令处于 EX 步骤并判断是否进行转移。在判断周期结束之前，转移判断 S46 的结果从 EX 模块 46 返回至多路转换器 62。因此，根据上述结果在多路转换器 62 处选定已经同时读出的用于转移目的地的指令 S68 或用于非转移目的地的指令 S87

这二者中的一个，并且，结束 IF 模块中处理。所选定的指令 S62 在锁存于图 4 所示寄存器 43 之后被输出给 ID 模块 44。

以下说明图 4 所示的 ID 模块 44。

如图 4 所示，ID 模块 44 具有一译码器 65 和一寄存器文件 66。

5       译码器 65 根据控制信号 S51b 对经由寄存器 43 自 IF 模块 42 输入的指令 S62 进行译码、生成多个用于执行指令的控制信号并将控制信号 S65 输出给控制器 51。同时，译码器访问寄存器文件 66 并读取要用于在 EX 模块 46 中进行处理的数据。读出的数据 S66 锁存在寄存器 45 内并输出给随后的 EX 模块 46。

10       还有，当对来自寄存器 43 的指令 S62 的译码结果表示该指令是一转移指令时，译码器 65 就将转移目的地的地址 S44a 输出给图 5 所示 IF 模块 42 中的地址寄存器 83 以便进行存储，并将标志〔1〕存入模块 42 中的标志寄存器 81。

结果，在随后的周期中，对 EX 模块 46 中的转移指令进行转移判断。  
15       同时，在 IF 模块 42 中同时读取出用于转移目的地的指令和用于非转移目的地的指令。

以下说明图 4 所示的 EX 模块 46。

EX 模块 46 包括：一 ALU（算法与逻辑单元）67，它用于执行算法处理；一转移判断电路 68；以及，一未示出的地址生成电路。

20       ALU67 根据控制信号 S51c 按照来自控制器 51 的译码结果用数据 S66 进行信号处理。

所述地址生成电路生成数据存储器 69 内的地址，而数据存储器 69 则存储有 ALU67 中处理结果的数据。

25       请注意，就算法处理而言，ALU68 使用存储在数据存储器 69 中并通过访问寄存器文件 66 输出给寄存器文件 66 的数据。

还有，ALU67 通过寄存器文件 66 将算法处理的结果存入存储器 69。

EX 模块 46 将 ALU68 所处理的算法结果以及地址生成电路所生成的地址通过寄存器 47 输出给 MEM 模块 48。

30       当在 ALU67 中执行的指令是转移指令时，转移判断电路 68 就将指示进行转移的转移判断结果 S46 输出给 IF 模块 42。同时，对转移条件求值，从而进行转移判断。IF 模块 42 根据转移判断的结果 S46 选择在图 5 所示的多路转换器 62 处同时取出的用于转移目的地的指令或用于

非转移目的地的指令这两者之中的一个。

以下说明 MEM 模块 48。

MEM 模块 48 具有数据存储器 69 和未示出的控制电路。

5 在接收到写操作指令时，MEM 模块 48 根据来自控制器 51 的控制信号 S51d 将自 EX 模块 46 输入的算法处理结果数据存至（写至）数据存储器 69 中通过寄存器 47 自 EX 模块 46 输入的地址。

在接收到读操作指令时，MEM 模块根据来自控制器 51 的控制信号 S51d 从数据存储器 69 中通过寄存器 47 自 EX 模块 46 输入的地址读取数据。

10 在接收到不需要访问数据存储器 69 的指令时，MEM 模块 48 通过寄存器 49 将通过寄存器 47 自 EX 模块输入的算法处理结果数据输出给 WB 模块 50。

此外，MEM 模块根据来自控制器 51 的控制信号在多路转换器处选定读自数据存储器 69 的数据或来自 EX 模块的算法处理结果数据这两者之  
15 中的一个，并将其通过寄存器 49 输出至 WB 模块 50。

以下说明 WB 模块 50。

WB 模块 50 根据控制信号 S51e 将通过寄存器 49 自 MEM 模块 48 输入的数据存至 ID 模块 44 中的寄存器文件 66。

以下说明处理器 41 的操作。

20 图 1 是处理器 41 中出现转移时转移指令的流水线处理的示意图。

首先，在周期“1”中将指令 n 锁存至图 4 中所示的 IF 模块 42，然后，在下一个周期“2”中于 ID 模块 44 处对指令 n 译码，同时，在 IF 模块 42 处取出指令 n+1。

这期间，在 IF 模块 42 中，将表示〔0〕的标志存入图 5 所示的标志寄存器 81，并且，访问控制器 84<sub>1</sub> 至 84<sub>8</sub> 根据程序计数据 60 所指示的存储在地址寄存器 82 中的地址从存储器 80 中读出指令，并通过多路转换器 86 和 62 将读出的指令输出给寄存器 43。  
25

而且，在 ID 模块 44 处将指令 n 确认为一转移指令，图 4 所示的译码器 65 将标志〔1〕存入图 5 所示的指令存储器 61 的标志寄存器 81 内，并且，用于转移目的地的指令的地址存在地址寄存器 83 内。  
30

然后，在图 1 所示的周期“3”中，在 EX 模块 46 的转移判断电路 68 处判断指令 n 是否满足转移条件。当满足所述条件时，将表示满足该条

件的转移判断结果 S46 输出给图 4 和图 5 所示的多路转换器 62。

同时，在图 5 所示的指令存储器 61 中，访问控制单元 84<sub>1</sub> 至 84<sub>8</sub> 根据存储在地址寄存器 82 和 83 内的地址从存储器 80 中读出用于转移目的地的指令  $\underline{m}$  和用于非转移目的地的指令  $n+2$ 。然后，将用于非转移目的地的指令  $n+1$  (S86) 和用于转移目的地的指令  $\underline{m}$  (S87) 输出给多路转换器 62，此后，根据转移判断的结果 S46 在多路转换器 62 处选出用于转移目的地的指令  $\underline{m}$ ，并将其作为指令 S62 通过寄存器 47 输出给 MEM 模块 48。

在周期“2”中，中断在 IF 模块 42 处取出的指令  $n+1$ 。

10 然后，在周期 4 中，MEM 模块 48、ID 模块 44 和 IF 模块 42 分别执行指令  $\underline{n}$  的 MEM 步骤、指令  $\underline{m}$  的 ID 步骤以及指令  $m+1$  的 IF 步骤。

此后，在周期 5 中，WB 模块 50、EX 模块 46、ID 模块 44 以及 IF 模块 42 分别执行指令  $\underline{n}$  的 WB 步骤、指令  $\underline{m}$  的 EX 步骤、指令  $m+1$  的 ID 步骤以及指令  $m+2$  的 IF 步骤。

15 然后，只要没有转移指令，就可顺序地执行指令  $m+3$ 、 $m+4$ 、……的 IF 步骤、ID 步骤、EX 步骤、MEM 步骤和 WB 步骤。

图 2 是处理器 41 中转移指令进行转移时的流水线处理的示意图。

在这种情况下，可在周期“1”和“2”中实现与上述因转移指令而出现转移的情况的流水线处理相同的处理。

20 然后，在周期“3”中，在 EX 模块的转移判断电路 68 中判断指令  $\underline{n}$  是否满足转移条件。在不满足上述条件时，将表示不满足转移条件的转移判断结果 S46 输出给图 4 和图 5 所示的多路转换器 62。

同时，在图 5 所示的指令存储器 61 中，根据存储在地址寄存器 82 和 83 中的地址，在访问控制单元 84<sub>1</sub> 至 84<sub>8</sub> 内从存储器 80 读出用于转移目的地的指令  $\underline{m}$  和用于非转移目的地的指令  $n+2$ 。然后，将用于非转移目的地的指令  $n+2$  (S86) 和用于转移目的地的指令  $\underline{m}$  (S87) 输出给多路转换器 62，其中，根据转移判断的结果 S46 选定用于非转移目的地的指令  $n+2$  并将其作为指令 S62 通过寄存器 47 输出给 MEM 模块 48。

而且，中断周期“2”中在 IF 模块 42 内取出的指令  $n+1$ 。

30 然后，在周期“4”中，MEM 模块 48、ID 模块 44 以及 IF 模块 42 分别执行指令  $\underline{n}$  的 MEM 步骤、指令  $n+1$  的 ID 步骤以及指令  $n+2$  的 IF 步骤。

此后，在周期“5”中，WB 模块 50、EX 模块 46、ID 模块 44 和 IF

模块 42 分别执行指令  $n$  的 WB 步骤、指令  $n+1$  的 EX 步骤、指令  $n+2$  的 ID 步骤以及指令  $n+3$  的 IF 步骤。

然后，只要没有转移指令，就指令  $n+4$ 、 $n+5$ 、……而言，可由 IF 步骤、ID 步骤、EX 步骤、MEM 步骤以及 WB 步骤顺序地实现相同的处理。

5 如上所述，依照处理器 41，当在 ID 模块 44 内把一指令确认为转移指令同时该转移指令正被执行并在下一个周期的 EX 模块 46 中进行转移判断时，一旦获得了转移判断的结果 S46，就同时读出用于转移目的地的指令和用于非转移目的地的指令，并且选出一适当的指令。结果，可以以与是否进行转移的转移判断结果 S46 无关的方式在下一周期将用于  
10 转移或非转移目的地的指令输出给 ID 模块 44。

所以，与相关技术的上述并行处理器 1 相比，在转移时可以有效地防止处理效率下降。

具体地说，依照处理器 41，与不预测转移的传统方法相比，在出现转移指令时，可将处理时间缩短若干个周期。

15 此外，依照处理器 41，与预测转移出现的传统方法相比，在预测被证明是错误的时，可将处理时间缩短若干个周期。

再者，依照处理器 41，与传统的延时转移方法相比，当用若干插入的空操作指令来执行转移指令以便防止延时段为其它指令所占用时，可以降低无用的时钟消耗（转移损失）。

20 本发明并不局限于上述实施例。

例如，在上述实施例中，可将具有单个读取端口的单个存储器端口用作图 5 所示的存储器 80，但是，也可以使用具有多个读取端口的多端口存储器。

而且，在上述实施例中，如图 4 所示那样说明了五步流水线处理的结果，但是，本发明也可用于多于五步的流水线处理。  
25

此外，对于图 4 所示的指令存储器 61 的结构，只要具有相同的功能就不局限于具体如图 5 所示的结构。

如上所述，依照本发明，可以有效地防止流水线处理的效率因转移指令而下降。

30

# 说明书附图

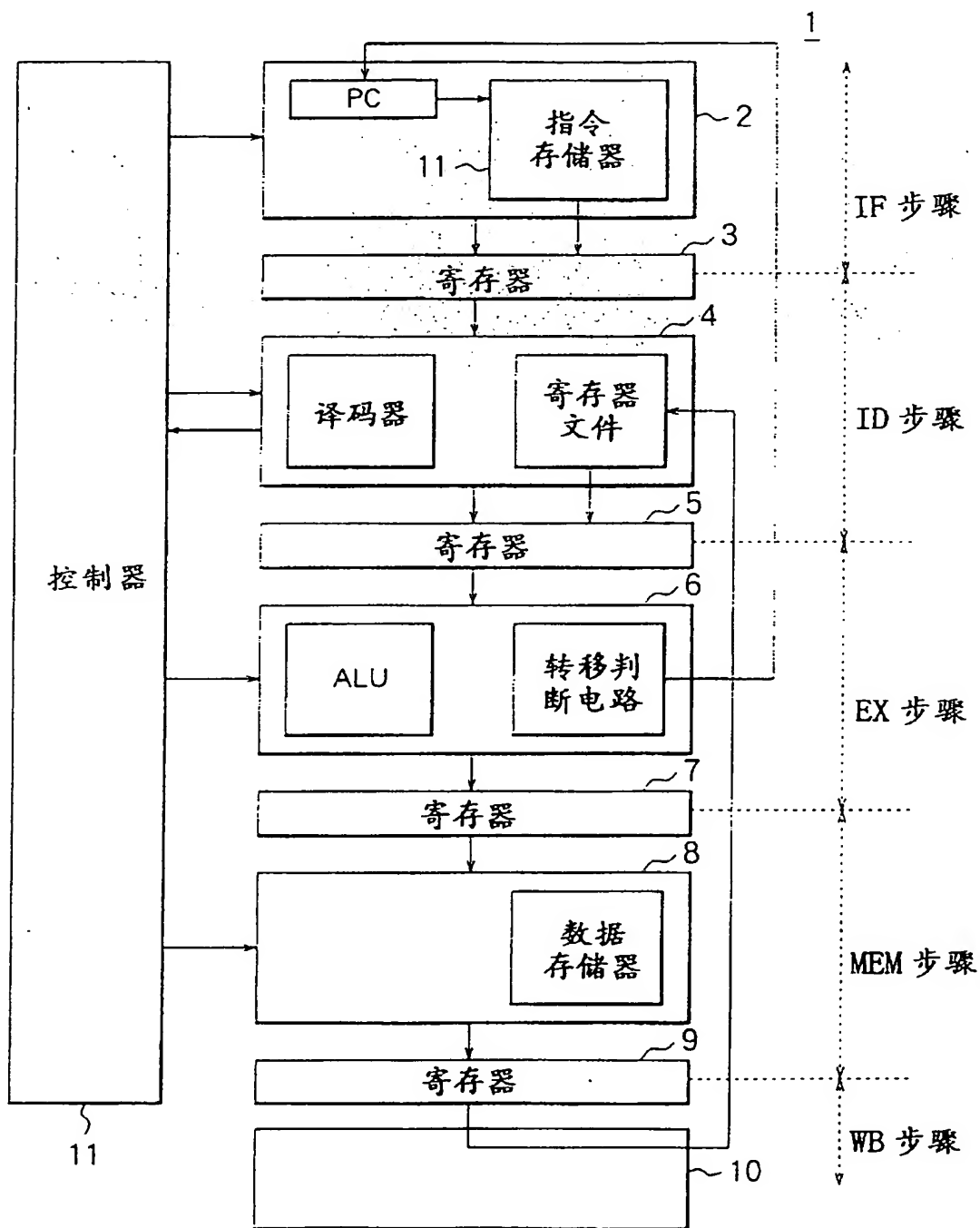


图 1

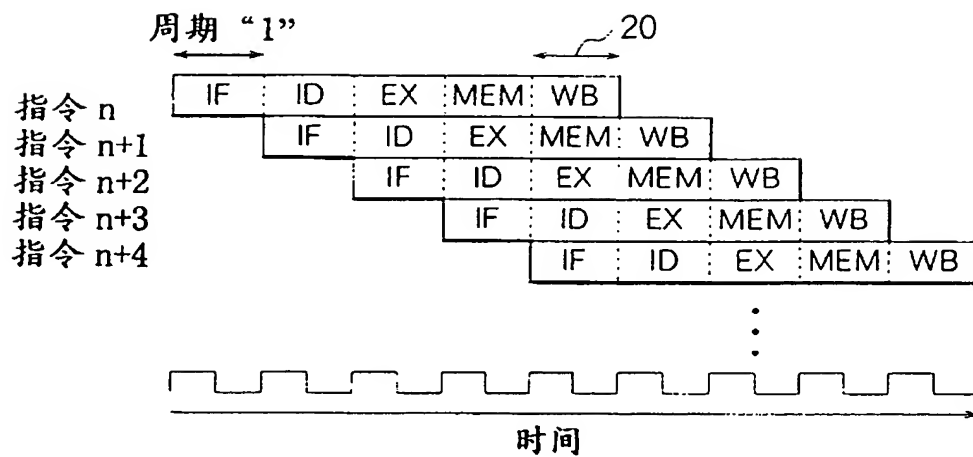


图 2

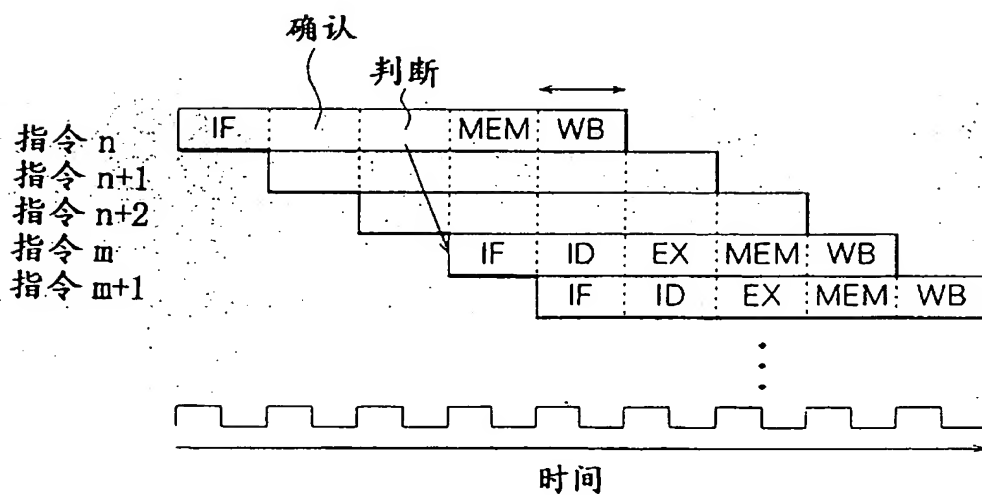


图 3

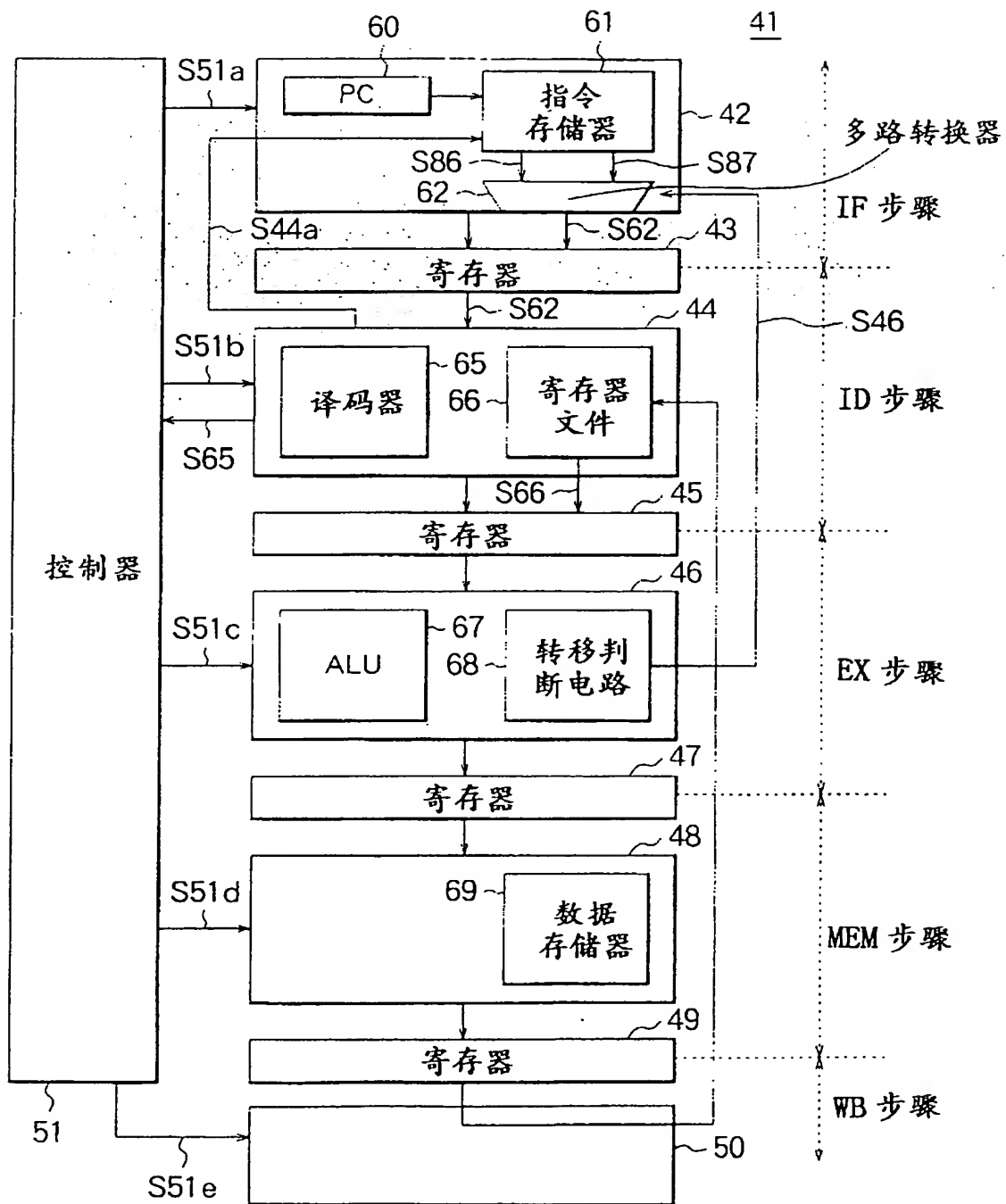
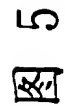


图 4





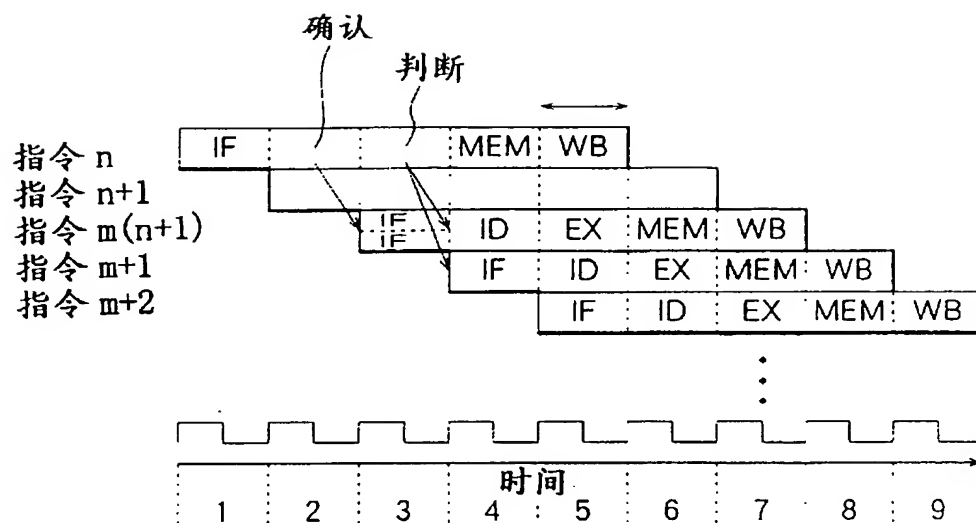


图 7

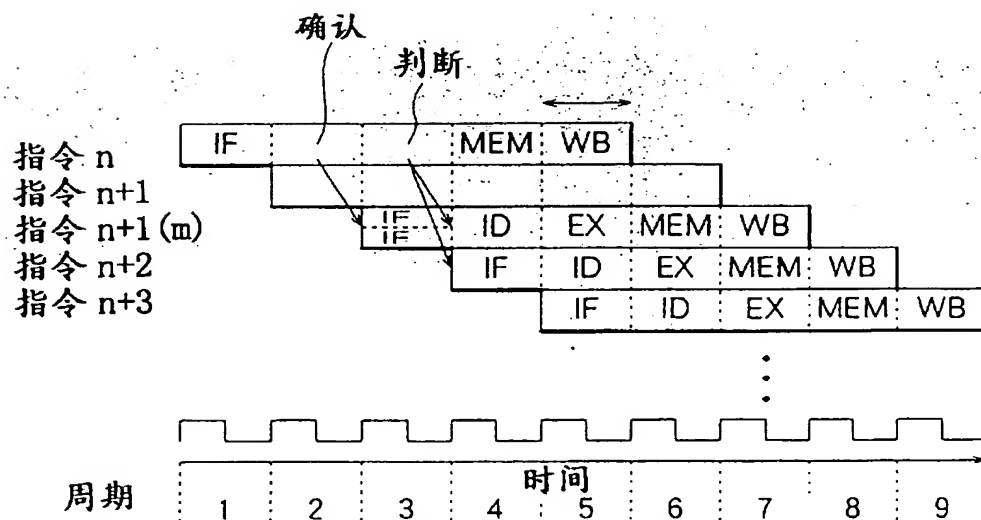


图 8